VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Vizualizace neuronových sítí

Visualization of Neural Networks

2013

Petr Feichtinger

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Petr Feichtinger**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Vizualizace neuronových sítí
Visualization of Neural Networks

Zásady pro vypracování:

Cílem bakalářské práce je získat přehled v oblasti grafové vizualizace, implementovat alespoň dvě různé metody a pomocí nich zobrazit neuronové sítě.

1. Nastudovat problematiku grafové vizualizace se zaměřením na typ neuronové sítě Self-Organizing Map.
2. Dle pokynů vedoucího naimplementovat vybrané vizualizace v programovacím jazyce C#.
3..Otestovat funkčnost implementací.

Seznam doporučené odborné literatury:

Elias Pampalk, Andreas Rauber, Dieter Merkl. Using Smoothed Data Histograms for Cluster Visualization in Self-Organizing Maps. In Proceedings of the Intl Conf on Artificial Neural Networks (ICANN 2002), , August 27.-30. 2002, Madrid, Spain.
Alfred Ultsch and H. Peter Siemon. Kohonen's self-organizing feature maps for exploratory data analysis. In Proceedings of the International Neural Network Conference (INNC'90). Kluwer, 1990.
Alfred Ultsch. Maps for the Visualization of high-dimensional Data Spaces. In Proceedings Workshop on Self-Organizing Maps (WSOM 2003), Kyushu, Japan, (2003).
Alfred Ultsch. U*-Matrix: a Tool to visualize Clusters in high dimensional Data, Technical Report No. 36, Dept. of Mathematics and Computer Science, University of Marburg, Germany, (2003).

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Lukáš Vojáček**

Datum zadání: 18.11.2011
Datum odevzdání: 07.05.2013

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.*

V Ostravě 7. května 2013 ..... Feichtinger .....

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2013 ..... Feichtinger .....

Rád bych na tomto místě poděkoval vedoucímu své bakalářské práce panu Ing. Lukáši Vojáčkovi za trpělivost, rady a připomínky k obsahu a formě zpracování.

Abstrakt

Bakalařská práce je zaměřena na vizualizaci neouronové sítě typu Self-organizing map. Zabývá se grafickým zobrazení organizované nuronové síťě za použití několika odlišných metod vizualizace dat.

Klíčová slova: Samoorganizující se mapy, SOM, neuronové sítě, U-Matrix, Vector Activity Histogram, Cluster Connections, Neighbourhood Graph, Vector Fields

Abstract

The bachelor thesis is focused on the type of neural network Self-organizing maps and her visualization. It deals with the visualization of organized neural networks using several different methods of data visualization.

Keywords: Self-organizing map, SOM, neural networks, U-Matrix, Vector Activity Histogram, Cluster Connections, Neighbourhood Graph, Vector Fields

Seznam použitých zkratek a symbolů

SOM	–	Samoorganizující se mapy
U-Matrix	–	Unified Distance Matrix
VAH	–	Vector Activity Histogram
CC	–	Cluster Connections
NG	–	Neighbourhood Graph
VF	–	Vector Fields
OOP	–	Objektově orientované programování
GUI	–	Grafické uživatelské rozhraní

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Práce se bude zabývat problematikou grafové vizualizace umělých neuronových sítí. Tato bakalářská práce je věnována umělé neuronové síti typu SOM, kde v první části seznamuji s problematikou neuronových sítí. Zde se zabývám nejen popisem, ale také historii těchto neuronových sítí.

Druhá část práce se zabývá SOM a jejim uplatněním v neuronových sítích. V této částí se zaměřím i na zobrazení SOM a metody vizualizace SOM.

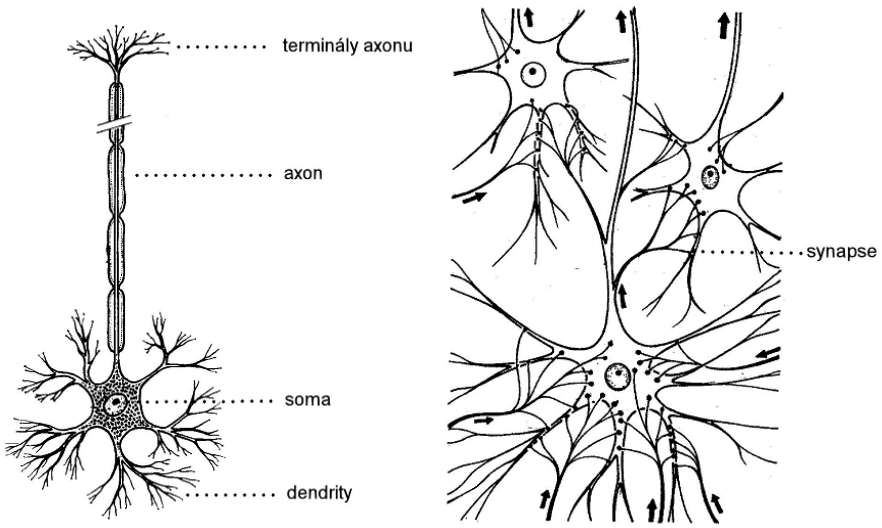
Ve třetí části práce jsou vysvětleny jednotlivé metody vizualizace sítě. Zde se pokusím použité metody, které jsem naimplementoval, popsat. Dále se zaměřím na výklad teoretických informací o metodách a jejich nasazení v SOM. Pak budou následovat ukázky z implementace metod, které budou doplněny grafickými výstupy z programu.

V předposlední kapitole se budu věnovat praktické částí této bakalářské práce.

2 Neuronové sítě

2.1 Biologický vzor umělých neuronových sítí

V prvé řadě je důležité se seznámit s motivem vzniku umělých neuronových sítí, s neurony a jejich reálným vzorem, kterým je lidský mozek. Tvorba neuronových sítí byla motivována snahou pochopit a zobrazit model mozku a vysvětlit způsob, jakým lidský mozek funguje. Informace z neurofyziologických poznatků umožnily vytvořit zjednodušené matematické modely, které se dají využít pro řešení praktických úloh z umělé inteligence. Poznatky z neurofyziologie sloužily jenom jako zdroj inspirace. Později navržené modely jsou již dále vyvíjeny bez ohledu na podobnost lidskému mozku. Bez ohledu na vývoj neuronových síti lze stále využívat poznatky z neurofyziologie, které mohou sloužit k inspiraci pro vývoj nových metod nebo jenom k matematickému popisu již existujících metod. Neuronové sítě jsou podobné těm v mozku. Skládají se ze soustavy navzájem propojených neuronů. Pro časovou a výpočetní složitost se matematické modely skládají jenom ze zlomků neuronů oproti lidskému mozku, který obsahuje průměrně 14 miliard neuronů, z nichž každý může být spojen s 5000 jinými neurony. Model z levé strany obrázku 1 (obrázek byl převzat z [1]) je struktůra biologického neuronu. Neuron kromě vlastního těla, tzv. *somatu*, má výstupní přenosové kanály, tzv. *axony*, z kterého odbočuje řada větví (*terminál axonu*), které se převážně dotýkají vstupu dalšího neuronu (*dendrity*).



Obrázek 1: Reálná neuronová síť

K přenosu informace slouží mezineuronové rozhraní známé jako *synapse*. V pravé části obrázku 1 je ukázka neuronové sítě v mozku.

2.2 Umělá neuronová síť

Umělá neuronová síť se skládá z formálních neuronů, které jsou vzájemně propojené. Propojení je vytvořeno tak, že jeden neuron může mít libovolný počet neuronů, ale má pouze jeden výstup.

Využití umělých neuronových sítí v různých oblastech výzkumu:

- Zpracování dat, řeči a obrazu
- Třídění dat
- Hledání v textu
- Matematické problémy
- Umělá inteligence
- Komprese dat

2.3 Historie umělých neuronových sítí

V článku „A logical calculus of the ideas immanent in nevrous activity"[2], který vyšel v roce 1943 v časopise „Bulletin of mathematical biophysics" vytvořili autoři Warren McCulloch a Walter Pitts jednoduchý matematický popis neuronů a ukázali, že neurony mohou fungovat dle pravidel Booleovy algebry. Dokázali, že po spojení takovýchto jednoduchých jednotek do neuronové sítě je možno vybudovat zařízení, které by bylo schopné samostatně provádět výpočetní operace nad Booleovou algebrou.

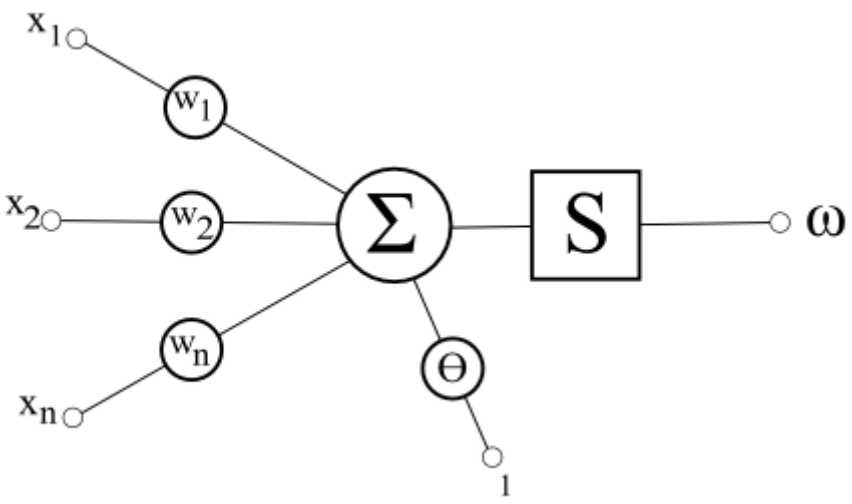
Na obrázku 2 je zobrazen návrh neuronu, který vytvořili Warren McCulloch a Walter Pitts. Z návrhu neuronu můžeme odvodit, že má určitý počet vstupů a jediný výstup. Z toho vyplývá, že jeho výstup může být vstupem jenom pro jeden neuron. Pak neuron obsahuje libovolnou funkci, kde vstupní hodnoty funkce jsou vstupní hodnoty neuronu a výstupem z neuronu je funkční hodnota funkce.

Neuron má n reálných vstupů $x_1, ..., x_n$. Vstupy jsou ohodnoceny reálnými synaptickými váhami $w_1, ..., w_n$, které určují jejich propustnost a slouží taky jako paměť neuronu. Obecně neuron můžeme popsat podle vztahu:

$$y = S\left(\sum_{i=1}^{N} w_i x_i + \Theta\right) \tag{2.1}$$

Kde:

- y - výstup neuronu
- S - nelineární přenosová funkce neuronu (Sigmoidální funkce, hyperbolický tangent, radiální báze)
- w_i - synaptické váhy
- x_i - vstupy neuronu
- Θ - práh

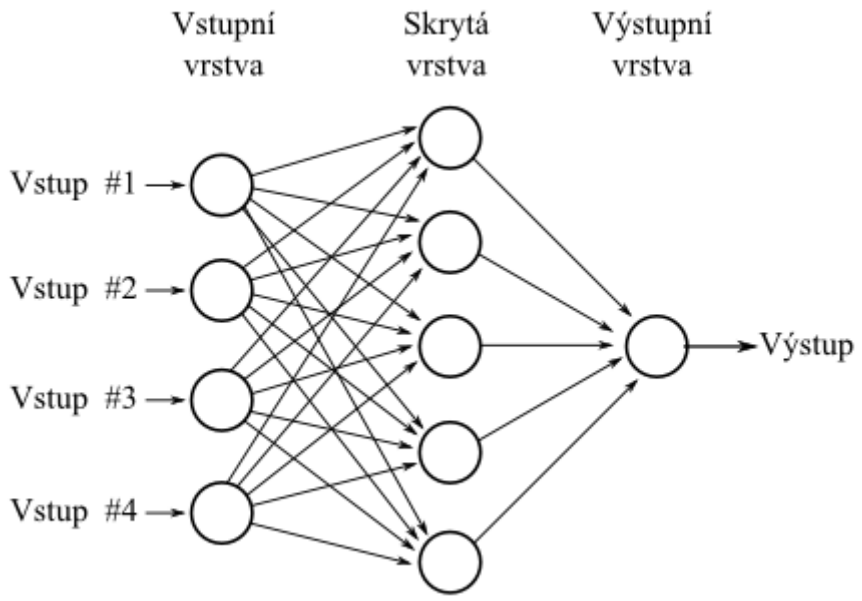


Obrázek 2: První navrhnutý model neuronu

Po spojení jednotlivých neuronů dostáváme neuronovou síť. Učení neuronové sítě se provádí změnou vah, tvarováním přenosové funkce, úpravou počtu neuronu v síti nebo i topologickým uspořádáním sítě.

2.4 Druhy neuronových sítí

Při učení s učitelem existuje vnější kritérium určující, který vstup je správný, v síti se nastavují zpětné vazby podle toho, jak blízko je výstup kritéria. Do nejjednodušší neuronové sítě patří Perceptron, jehož model je identický se základním modelem umělého neuronu. Jedná se o jednovrstvou síť, která využívá znaménkovou aktivační funkci či jednotkový skok. Implementace je jednoduchá, s tím souvisí jeji omezené použití. Nejčastěji se používá jako klasifikátor pro lineárně separovatelné obrazy. Po spojení Perceptronů do více vrstev, dostaneme složitější strukturu sítě, která se označuje jako Back Propagation. Používá zejména sigmoidální funkci nebo hyperbolický tangens. Vzhledem k častému používání vícevrstvého perceptronu a jeho určitým nedostatkům existuje mnoho variant toho modelu, který se snaží zlepšit jeho vlastnosti. Nejčastější je použití při klasifikaci obrazů, aproximací funkcí nebo predixe časových řádů.



Obrázek 3: Back Propagation

Učení bez učitele nepoužívá žádná vnější kritéria a mezi nejznámější druhy neuronových sítí bez učitele patří Kohonenové mapy, známé také jako samoorganizující se mapy (SOM) , což je dvouvrstvá síť s dopředným šířením. První vrstva udává dimenzi vstupních dat a druhá (výstupní) vrstva je uspořádána do určité topologické struktury. Používá se při analýze dat, shlukování či vytváření sémantických map.

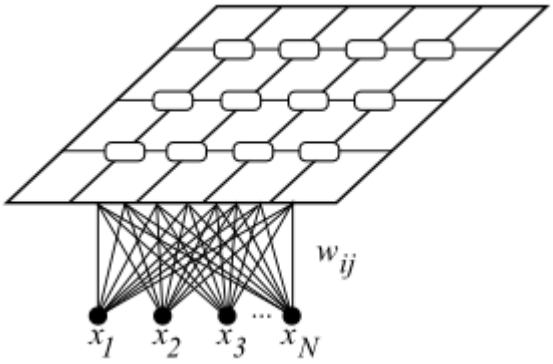
3 Samoorganizující mapa

3.1 Kohonenova síť

Jedná se o první samoorganizující mapu, kterou popsal profesor Teuvo Kohonen[3] v letech 1981-1982, proto bývá často označována jako Kohonenova síť. Jedná se o základní samoorganizující mapu, která je plně propojena neuronovou sítí s topologicky uspořádanou výstupní vrstvou. SOM využívá funkce sousedství, ktera má na chování mapy zásadní vliv. Princip sousedství spočívá v tom, že zaručuje uspořádanost naučené mapy tak, že blízká data ve vstupním prostoru aktivují neurony, které jsou si blízké ve výstupním prostoru. Pro zachování této funkce musí byt nenulové okolí.

Funkce sousedství udává poloměr okolí $\boldsymbol{p}$ *od daného neuronu, všechny neurony v tomto okolí jsou označeny jako sousedé onoho neuronu. Okolí může být ostré, tedy bud' je neuron sousedem, nebo není, nebo spojité, tedy každému neuronu je přiřazena hodnota, která určuje jeho náležitost do okolí pomoci funkce příslušnosti. Tou bývá obvykle Gaussova funkce, nebo Mexický klobouk (angl. mexican hat).* [5]

Získané informace o souvislostech mezi daty z naučené Kohonenovy mapy jsou velice abstraktní. Máme však způsoby pro prezentování informací naučené Kohonenovy mapy a jedním z nejvyužívanějších je grafová vizualizace.



Obrázek 4: Příklad jednoduché Kohonenové sítě

- Vstupní vektor:

$$X = [x_1, ..., x_N]^T, x_i \in \mathbf{R} \tag{3.1}$$

- Výstupní hodnota neuronů je definována jako vzdálenost mezi vstupním a váhovým vektorem:

$$y = \sum_{i=1}^{N} \left(x_i\left(t\right) - w_i\left(t\right)\right)^2 \tag{3.2}$$

  Kde:

  - y - výstup z neuronu
  - $x_i\left(t\right)$ - jednotlivé elementy vstupního vzoru
  - $w_i\left(t\right)$ - odpovídající váhy neuronu

3.2 Grafová vizualizace SOM

Pomocí SOM vytváříme podobnostní graf vstupních dat. Zde převádíme nelineární vztahy mezi vysoce-dimenzionálními daty do jednodušších geometrických vztahů (obrazových bodů v nízko-dimenzionálním zobrazení) obvykle do podoby pravidelné dvojrozměrné mapy uzlů. Grafová vizualizace SOM vychází z teorie grafů, proto je nutné si ujasnit některé pojmy a jejich význam a souvislosti mezi nimi v SOM:

- *datová oblast:* $D \subset R^n$: podmnožina R^n, ve které můžeme pozorovat datové body
- *vstupní data:*

$$E = \{x_1, ..., x_d\} \quad \text{pro } x_i \in D \tag{3.3}$$

- *vzdálenost dat:* měrná vzdálenost definována v datovém prostoru

$$D \times D \rightarrow R^+ : d_{xy} = d\left(x, y\right) \geq 0, d_{ij} \text{ je zkratka pro } d\left(x_i, x_j\right) \tag{3.4}$$

- *neurony:*

$$M = \{n_1, ..., n_k\} \tag{3.5}$$

- *váha:* každý neuron je spojen s (vysoce-dimenzionálním) váhovým vektorem

$$w_i = \text{ váha } \left(n_i\right) \in D \tag{3.6}$$

- *váhová oblast:*

$$W = \{w_i, ..., w_n\} \tag{3.7}$$

- *mapová oblast:* $K \subset R^m, m \leq n$ v m-dimensionální oblasti s měrnou vzdálenosti

$$k : K \times K \to R^+ : k_{ij} = k\left(pos\left(n_i\right), pos\left(n_j\right)\right) \geq 0 \tag{3.8}$$

- *pozice neuronu:* každý neuron n_i má svojí pozici, tj. vektor souřadnic

$$pos_i = pos\left(n_i\right) \in K \tag{3.9}$$

- *funkce sousedství:* mapujeme

$$M \times M \times R^+ \to [-1, 1], h_{ij}\left(r\right) = h\left(n_i, n_j, r\right) \tag{3.10}$$

  s následujicími parametry:

  - $h\left(n_i, n_j, r\right) \geq h\left(n_i, n_j, r\right) \forall j \neq i$ pro $k_{ij} > 0$ a $r > 0$
  - $h\left(n_i, n_j, r\right) = 0$ pro $k_{ij} > r$
  - kde r je radius sousedství

- sousedství:

$$N_i = N\left(n_i\right) = \{n_j \in M \mid h_{ij}\left(r\right) \neq 0\} \tag{3.11}$$

  sada neuron s nenulovou funkci sousedství h

4 Metody vizualizace

4.1 Vector Activity Histogram

4.1.1 Vector Activity Histogram - popis metody

Tato metoda zobrazuje vliv uživatelem zvoleného vstupního neuronu. Neuron, který jsme zvolili, slouží jako základ pro výpočet vzdálenosti mezi váhami neuronů. Vzdálenost se počítá pomoci vzorce pro výpočet Euklidovské vzdálenosti mezi dvěma neurony, kde jeden neuron je ten, který jsme zvolili jako vstupní neuron, a jako druhý neuron dosazujeme neuron, který postupně vybíráme z mapy, až do té doby, než získáme celou matici vzdáleností. Hodnoty matice vzdálenosti pak přímo znázorníme v odstínech šedi, kde jednotlivé shluky mají jiné odstíny šedi. Metoda není vhodná pro podrobný popis naučené SOM, neboť shluky neuronů jsou znázorněny velmi všeobecně a není snadné pozorovat vazby mezi neurony uvnitř shluku.

Na obrázku 5 můžeme znatelně rozpoznat shluky podobných neuronů.

Obrázek 5: Vector Activity Histogram

Krok	Popis
1	Ze seznamu všech nouronů vyberu požadovaný neuron
2	Postupně vybírám neurony z mapy
3	Vypočítám Euklidovskou vzdálenost aktualního neuronu s námi vybraným neuronem
4	Normalizace součtů
5	Vykreslení normalizovaných součtů do grafu podle předem dané barevné palety

Tabulka 1: Popis algoritmu Vector Activity Histogram

4.1.2 Euklidovská vzdálenost

Pro výpočty vzdáleností mezi neurony se používá Euklidovská vzdálenost[4] . Jde o „obyčejnou“ vzdálenost mezi dvěma body, která je odvozena od vzorce pro Pythagorovou větu. Neuronová síť může být n-dimenzionální, a proto se používá vzorec Euklidovské vzdálenosti v n-dimenzionálním prostoru, kde máme definované vektory $\mathbf{p}$ a $\mathbf{q}$ jako

$$\mathbf{p} = (p_1, p_2, ..., p_n) \tag{4.1}$$

$$\mathbf{q} = (q_1, q_2, ..., q_n) \tag{4.2}$$

a vztahem pro Euklidovskou vzdálenost danou vzorcem:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{4.3}$$

Ukázka výpočtu Euklidovské vzdálenosti, jako parametry funkce jsou vektory vah neuronů.

```
private double Euklid(double[] p, double[] q)
{
    double dPom = 0;

    for (int i = 0; i < pocetVah; i++)
    {
        dPom += Math.Pow((p[i] − q[i]), 2);
    }

    return Math.Sqrt(dPom);
}
```

Výpis 1: Funkce výpočtu Euklidovské vzdálenosti

4.2 U-Matrix

4.2.1 U-Matrix - popis metody

Jedná se o velice často používanou metodu vizualizace pro prezentaci SOM.Vizualizace zobrazuje vzdálenost ve vstupním prostoru mezi váhami neuronů, které jsou vedle sebe. Metoda U-Matrix [6] může být zobrazována dvěma různými způsoby. První způsob je

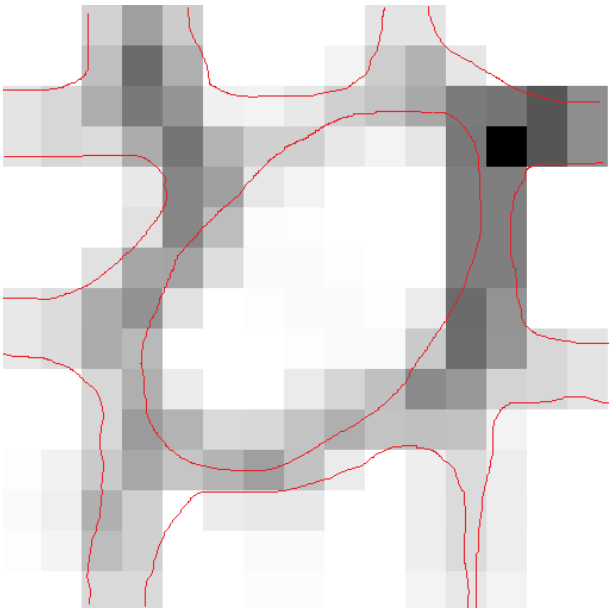
inter-node vzdálenost, který vyžaduje vložit do mapy další cesty mezi neurony. Druhý způsob zobrazuje jenom průměrnou hodnotu, u původních pozic uzlů jsou přiřazeny průměrné hodnoty vzdálenosti. Metoda U-Matrix je vhodná pro identifikaci možných interpolací mezi blízkými a odlehlými uzly.

U-Matrix zobrazuje neuronovou síť s barevně vyjádřenou informací o tom, jak moc se daný neuron liší od okolních ve svých váhách. Čím je světlejší, tím je jim podobnější. Světlé oblasti vymezené tmavší hranicí tak představují shluky s podobnou vzdáleností neuronů. Na obrázku 6 můžeme znatelně rozpoznat takovéto shluky neuronů (vyznačené červenou barvou).

Uvažujme neuron ***j*** *a všechny neurony, které sním sousedí označme jako množinu* ***NN(j)*** *. Pak:*

$$U_h(j) = \sum_{m \in NN(j)} d\left[w(j) - w(m)\right] \tag{4.4}$$

Kde $w(j)$ *je vektor vah neuronu* ***j****, a* $d\left[w(j) - w(m)\right]$ *je Euklidovská vzdálenost neuronu* ***l*** *od neuronu* ***m****, jenž přímo sousedí s neuronem* ***j****. Sečtena hodnota pro všechny neurony z množiny NN(j) udává celkovou vzdálenost neuronu* ***j*** *od jeho okolních neuronů. Tento výpočet provedeme pro všechny neurony, čímž získáme kompletní matici vzdáleností.* [5]



Obrázek 6: U-Matrix

Krok	Popis
1	Vybrání nueronu z matice
2	Výpočet Euklidovské vzdálenosti s okolními neurony
3	Sečtení těchto vzdáleností v okolí vybraného neuronu
4	Normalizace součtů
5	Vykreslení neuronu do grafu podle předem dané barevné palety
6	Když jsme neprošli celou matici, tak přejdem na krok 1

Tabulka 2: Popis algoritmu U-Matrix

4.3 Cluster Connections

4.3.1 Cluster Connections - popis metody

CC [7] je metoda vizualizace, která pro výpočet potřebuje informace obsažené ve váhách vektorů již naučené SOM. Dalo by se říct, že vzdálenosti mezi váhami vektorů sousedních neuronů slouží k určení, zda tyto neurony patří k stejnému shluku. Pokud patří do stejného shluku, měly by být neurony spojené, pokud jsou příliš vzdáleny, spojeny nebudou.

Pro zlepšení vizualizace je při výpočtech použita sada prahových hodnot, která slouží k vizualizaci různé míry podobnosti. V důsledku toho technika vizualizace je schopna odhalit mimoshlukovou spojitost výstupních hodnot, a tak zobrazovat informační kvalitu mezi neurony v jednom shluku.

Na základě analýzy vzdálenosti mezi párem sousedních neuronů a stanovením limitních hodnot, můžeme odvodit a přidat další spojení o jiné intenzitě. Další spojení můžou zobrazovat různou míru podobnosti v rámci shluku, ale také vzájemné podobnosti celého seskupení. Různé implementace této základní myšlenky mohou sloužit k vizualizaci výsledné mapy v různých možných uživatelských rozhraních, z nichž některé jsou uvedeny níže.

1. Pevná sada dvou nebo více prahových hodnot, které můžou být zadané bud' automaticky, na základě průměrné, minimální a maximální vzdálenosti mezi sousedními váhami vektorů, nebo ručně po analýze vzdálenosti k zobrazení spojení s různou intenzitou.

2. Druhem „posuvníku“, který znázorňuje vzdálenostní přímku, pak vizualizace zobrazuje vzdálenosti různých dvojic neuronů, což uživateli umožňuje rozhodnout se,

do jaké vzdálenosti budou neurony propojeny. Tak může uživatel pozorovat sadu spojených neuronů. Pomocí „posuvníku“ může uživatel interaktivně zmenšovat nebo zvětšovat hranici spojitostí neuronů, a tak dovoluje hlubší náhled do struktury vstupních dat.

3. Všechny neurony mohou být spojené. Na základě vzdálenosti mezi neurony určíme barvu spojení; tj. vzdálenosti jsou přímo převedeny na barvy nebo různé odstíny šedi, od černé, která znázorňuje téměř totožné neurony, až po bílou, jež označuje velmi vzdálené neurony.

4.3.2 Aplikování metody

Výsledek vizualizace z obrázku 7 vychází z nastavení tří různých prahových hodnot. Tento typ vizualizace, která využívá tři prahové hodnoty, jsem vybral z důvodu použití šedé škály barev a myslím, že trojice prahových hodnot dostatečně znázorňuje funkčnost CC vizualizace.

Obrázek 7: Cluster Connections

Na obrázku 7 vidíme výsledek metody CC, kde můžeme pozorovat dvoje různé nastavení prahových hodnot. Na obrázku 7 (a) pozorujeme černou barvu spoje mezi sousedními uzly, což nám říká, že vzdálenost jejich hmotností je menší než prahová hodnota *t1*. Tmavě šedá spojka mezi sousedními uzly nám zobrazuje, že vzdálenost je mezi prahovými hodnotami *t1* a *t2*. Světle šedá barva zobrazuje vzdálenost mezi prahovými

hodnotami *t2* a *t3*. Bílá barva je použita pro vzdálenost větší než je prahová hodnota *t3*. Při porovnání grafů (a) a (b) na obrázku 7 můžeme pozorovat, rozdíl ve vstupních prahových hodnotách, a to takový, že při nastavení *t1* = *t2* z výsledné vizualizace zmizela tmavě šedá barva.

4.4 Vector Fields

V této sekci si popíšeme dvě vizualizační metody, které jsou založené na technice vektorových polí. První metoda je označována jako Gradient Field [8], kde gradient je vykreslen jako vektor v 2D mapě, potom každý uzel v SOM map je zobrazen výhradně pomoci jednoho vektoru. Umístění centra shluku nám indikují vektory, které mají směr natočení orientovaný právě na centrum shluku. Celistvost vektorů pak vytváří vyhlazené vektorové pole. Druhá metoda založena na vektorových polích je označována jako Borderline vizualization. Metoda je odvozena od Gradient Field a poskytuje další pohled na shluky a jejich ohraničení.

4.4.1 Gradient field

Každý vektor a_i je počítán na základě vektorových prototypů, topologii mapy a sousedních uzlů, který pak bude umístěn na odpovídající místo v mapě, které označíme jako ξ_i. Vektor a_i se skládá ze dvou komponentů u a v, a_i^u a a_i^v, které reprezentují vertikální a horizontální souřadnice vektoru a_i. Konečná podoba výstupního vektoru se skládá ze dvou hlavních kroků. Prvním z nich je výpočet vzdálenosti vah mezi ostatními vektorovými prototypy, kde se spočítají odděleně osy u a v v kladném a záporném směru. Ve druhém kroku jsou jednotlivé výpočty spojeny pro každou souřadnici a pak normalizovány tak, aby se zabránilo kolizi vektorů.

Prvně je třeba objasnit propojení vektoru mezi pozici ξ_i a ξ_j ve výstupní mapě. Kde

$$x_{ij} = \overrightarrow{\xi_i \xi_j} = \xi_j - \xi_i \tag{4.5}$$

a úhel orientace vektoru $x_{ij} : \alpha$

$$\alpha = arctan\left(\frac{x_{ij}^v}{x_{ij}^u}\right) \tag{4.6}$$

Vztahy použijeme na sousední uzly a jejich vzdálenosti (mezi uzly ξ_i a ξ_j s délkou x_{ij}), a pak můžeme váhu rozdělit na u a v orientaci pomoci:

$$\omega_{ij}^u = cos\alpha \cdot h_\sigma\left(\|x_{ij}\|\right) \tag{4.7}$$

$$\omega_{ij}^{v} = sin\alpha \cdot h_{\sigma}\left(\|x_{ij}\|\right) \tag{4.8}$$

Hodnota ω_{ij}^{u} bude blízko nule, když kterákoliv vzdálenost mezi ξ_i a ξ_j bude vysoká. Ve výsledku bude mít uzel velmi malou hodnotu nebo bude $\xi_i^u = \xi_j^u$, tj. ξ_i je přímo nebo pod ξ_j se skoro žádným horizontálním posunem.

Hodnota σ má vliv na výsledek, tím pádem hodnota ω bude větší v závislosti na hodnotě σ, protože vysoká hodnota σ má tendenci zatěžovat vzdálené jednotky na mapě více než nízká hodnota σ.

Můžeme pozorovat, že ω_{ij}^{u} bude nabývat negativních hodnot v případě, že ξ_j bude na levé straně od ξ_i.

Následně si stanovíme vzorec pro souřadnici u (pro souřadnici v je vzorec obdobný). Vzdáleností mezi prototypy vektorů m_i a m_j, kde bereme v úvahu vážení těchto vzdálenost (označme ji jako ω), a jejich přiřazení buď k pozitivní nebo negativní straně souřadnice u :

$$\xi_{ij}^{u+} = \begin{cases} d_i(m_i, m_j) \cdot \omega_{ij}^{u} \\ 0 \end{cases} \quad \text{pro } \omega_{ij}^{u} > 0 \tag{4.9}$$

$$\xi_{ij}^{u-} = \begin{cases} d_i(m_i, m_j) \cdot \left(-\omega_{ij}^{u}\right) \\ 0 \end{cases} \quad \text{pro } \omega_{ij}^{u} < 0 \tag{4.10}$$

Rozdělení vážené vzdálenosti u vektorových prototypů na záporný a kladný směr napomáhá k nalezení směru daného vektoru m_i, a nakonec zobrazí, kde vektor a_i směřuje. Například když ξ_j je po pravé straně od sousedního ξ_i a vzdálenost mezi vektorovými prototypy je velká, hodnota ξ_{ij}^{u+}bude vysoká a významně přispěje k tomu, že vektor a_i bude směřovat doleva, dál od ξ_j.

Opakováním výpočtů pro všechny ξ_j dostaneme součet všech hodnot ξ_{ij}^{u+} a ξ_{ij}^{u-}. Součet označme ρ_i^{u+} a ρ_i^{u-} a vzorce budou vypadat následovně (pro v bude vzorec vypadat obdobně):

$$\rho_i^{u+} = \sum_{j=1}^{M} \xi_{ij}^{u+} \quad \text{pro } j \neq i \tag{4.11}$$

$$\rho_i^{u-} = \sum_{j=1}^{M} \xi_{ij}^{u-} \quad \text{pro } j \neq i \tag{4.12}$$

Když známe ρ_i^{u+} a ρ_i^{u-}, můžeme říci, do jaké míry jeden ze směru převažuje druhý směr, tj. pro $\rho_i^{u+} > \rho_i^{u-}$, když jsou sečtené vzdálenosti na pravé straně větší než ty

na levé straně, vektor bude směrovat nalevo. Když se přibližujeme k okrajům mapy, výsledné vektory bývají zpravidla menší, a to proto, že u okrajů mapy ubývá vektorových prototypů, které bychom mohli začlenit do výpočtu. Ubývání je zapříčiněno tím, že za hranicemi mapy už žádné prototypy nejsou. Výsledné vektory jsou zkreslené a většinou směřují za hranici výsledné mapy. Aby se tomu zabránilo, musí být provedena normalizace ω_{ij}^{u} hodnot pro kladný a záporný směr.

$$\omega_{ij}^{u+} = \sum_{j=1}^{M} \begin{cases} \omega_{ij}^{u} \\ 0 \end{cases} \quad \text{pro } \omega_{ij}^{u} > 0, j = i \tag{4.13}$$

$$\omega_{ij}^{u-} = \sum_{j=1}^{M} \begin{cases} -\omega_{ij}^{u} \\ 0 \end{cases} \quad \text{pro } \omega_{ij}^{u} < 0, j = i \tag{4.14}$$

Ve výsledku můžeme určit jednotlivé části a_i:

$$a_i^u = \frac{\rho_i^{u-} \cdot \omega_i^{u+} + \rho_i^{u+} \cdot \omega_i^{u-}}{\rho_i^{u+} + \rho_i^{u-}} \tag{4.15}$$

Kde sečtené rozdíly vstupního prostoru ρ_i^{u+} a ρ_i^{u-} jsou váženy proti příslušným součtum hodnot ω_i^{u+} a ω_i^{u-} . V případě, že uzel ξ_i neleží v blízkosti okraje u směru a ω_i^{u+} a ω_i^{u-} se budou rovnat, nebude to mít ve výsledku normalizace žádný efekt.

Pro výsledný vektor jsou nejdůležitější dílčí výpočtyρ_i^{u+} a ρ_i^{u-} a výsledný výpočet a_i. Zde může nastat několik situací:

- $\rho_i^+ \approx \rho_i^-$ – vzdálenosti jsou vyvážené a složky a budou malé. Když jsou ρ_i^+ a ρ_i^- malé, vektorový prototyp m_j z okolních uzlů ξ_i velmi podobný k m_i, a tim pádem se jedná o vektory v centru shluku. V případě, že oba ρ_i^+ a ρ_i^- jsou velké, je pravděpodobné, že ξ_i bude přímo mezi dvěma shluky a k žádnému z nich nebude směřovat. Takové jednotky se nazývají „interpolační“ a jsou snadno rozpoznatelné jako vektory sousedních uzlů, které směřují pryč od nich.

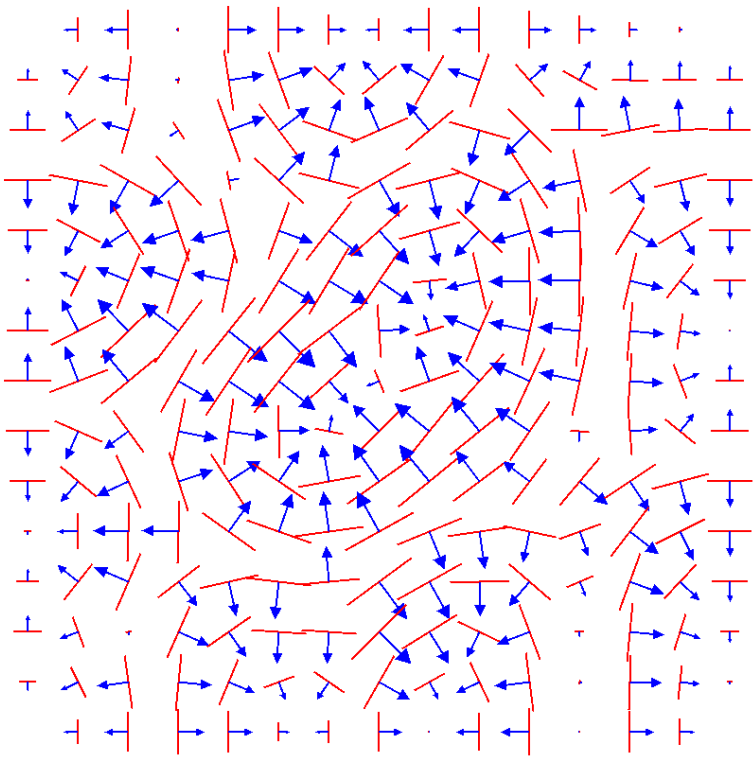
- $\rho_i^+ > \rho_i^-$– vzdálenosti v kladném směru převyšují vzdálenosti v negativním směru. m_i je více podobný k sousedním uzlům v záporném směru, a tak a_i bude poukazovat tímto směrem.

- délka a_i je určena rozdílem mezi ρ_i^+ a ρ_i^-. Čím jsou větší odlišnosti mezi nimi, tím větší bude výsledný vektor.

4.4.2 Borderline vizualization

Druhá metoda, založená na vektorových polích, klade důraz na znázornění hranic shluků. Jedná se o metodu odvozenou od Gradient field. Což znamená, že vedeme úsečku kolmo k vektoru a_i na každou stranu od začátku vektoru. Délka úsečky je přímo závislá na velikosti vektoru a_i, kde velikost vektoru je také i délkou úsečky.

4.4.3 Výsledek implementace



Obrázek 8: Vector Fields

Na obrázku 8 můžeme pozorovat vizualizaci Vektor Fields, kde jsme použili pro vstupní data naučenou SOM o velikosti 15x15. Ve výsledné vizualizaci jsou použity obě dílčí metody, jak Gradient Field tak Borderline Vizualization. Pomocí Gradient Field jsou patrné orientace jednotlivých uzlů (znázorněny jako modré šipky), které míří do center jednotlivých shluků. V mapě lze nalézt i interpolační uzly, které jsou malé a nejsou orientovány ke konkrétnímu shluku. Pomoci metody Berderlines Vizualization je lépe znázorněna hranice jednotlivých uzlů, ve kterých je použita červená úsečka kolmá

k vektoru. Když vynecháme interpolační uzly, můžeme říci, že velikost většiny vektorů je přímo úměrná vzdálenosti od centra shluku.

4.5 Neighbourhood Graph

4.5.1 Neighbourhood Graph - popis metody

Cílem metody vizualizace je získání sady hran, které spojují jednotlivé uzly podle předem daných kritérií. Kritéria určují, které sousední uzly budou propojeny v závislosti na vzdálenosti mezi jednotlivými uzly.

Výsledku NG [9] vizualizace můžeme dosáhnout pomocí dvou různých postupů výpočtu hran. V prvním postupu budeme potřebovat parametr r a definování uzlu okolo x_i jako x_j, který leží v oblasti o poloměru r a v jejím centru leží uzel x_i. Položky v matici sousednosti $N \times N$ jsou počítány jako :

$$e_{ij}^{rad} = \begin{cases} 1, i \neq j \wedge d(x_i, x_j) \leq r \\ 0 \end{cases} \tag{4.16}$$

Výsledný graf je neorientovaný kvůli symetrické metrické vzdálenosti d. Poloměr r slouží jako prahová hodnota a počet hran stoupá v závislosti na zvyšujícím se poloměru r.

V druhém postupu pro výpočet hran mezi sousedními uzly ve výsledné mapě budeme potřebovat číselný parametr k. Parametr určuje kolik sousedů máme spojit pomocí hran. Uzel x_j bude propojen s uzlem x_i tehdy, když bude uzel x_j patřit do nejbližšího sousedství $Nk(xi)$ uzlu x_i dané parametrem k, formálně:

$$x_j \in N_k(x_i) \Leftrightarrow \text{ Počet } \{x_l \in X : l \neq i, j \wedge d(x_l, x_i) < d(x_j, x_i)\} < k \tag{4.17}$$

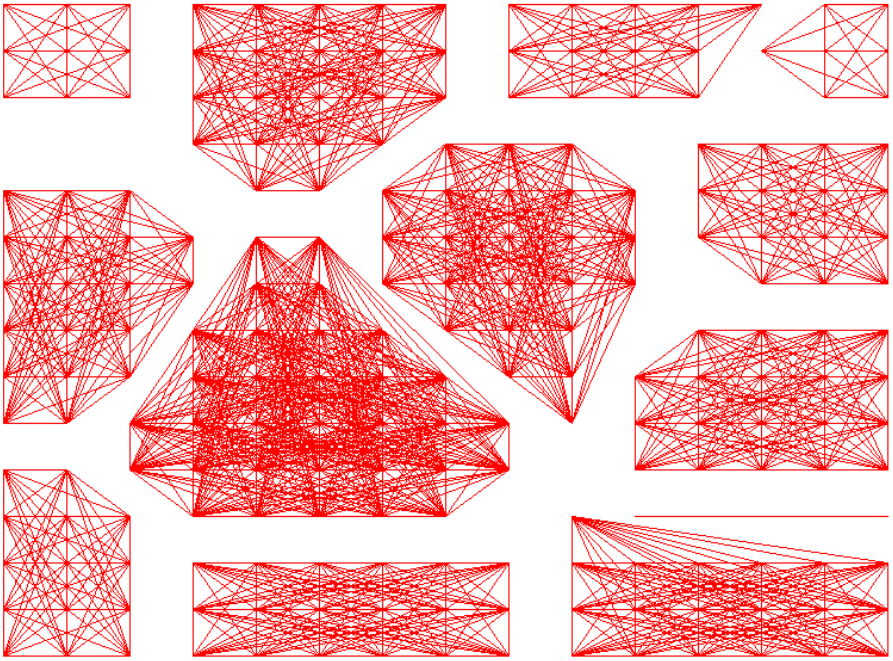
Kde „Počet“ označuje počet prvků množiny. Vztahy mezi nejbližšími sousedy nemusí být nutně symetrické, takže definice prvků v matici sousednosti budou definovány takto:

$$e_{ij}^{kNN} = \begin{cases} 1, i \neq j \wedge (x_i \in N_k(x_j) \vee x_j \in N_k(x_i)) \\ 0 \end{cases} \tag{4.18}$$

V tomto případě jsou hrany definovány, pokud x_i je k nejbližší soused k x_j nebo naopak. Stejně tak jako v prvním postupu navyšující se parametr k vede k navýšení počtu hran.

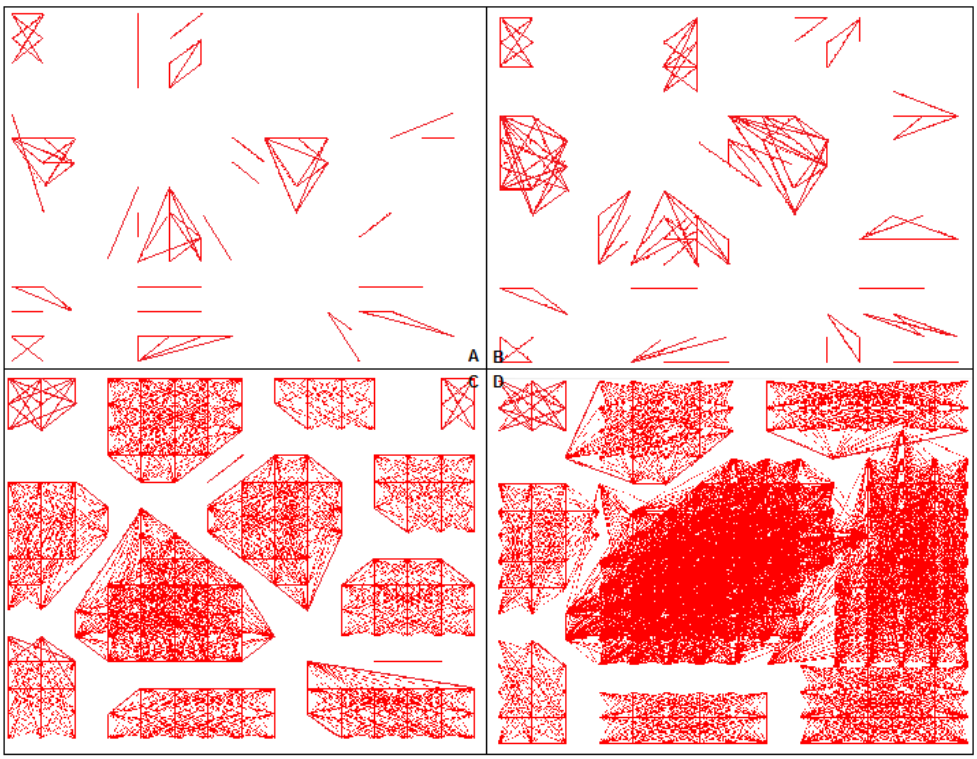
Výsledky výpočtů můžeme znázornit v dvou-dimenzionální mapě, v které zobrazíme vypočtené hrany pomoci úsečky spojující uzly mezi sebou. Výsledná mapa ukazuje, které oblasti SOM tvoří větší sousedství, kde leží interpolační uzly a jednotlivé shluky. Výsledná interpretace vizualizace NG závisí na velikosti mapy a zvolených parametrech r nebo k.

4.5.2 Výsledek implementace



Obrázek 9: Neighbourhood Graph

Na obrázku 9 lze pozorovat vizualizaci NG, kde jsme použili pro vstupní data naučenou SOM o velikosti 15x15. Ve výsledné vizualizaci jsem použil první postup výpočtu hran, a to že uživatel si zvolí oblast sousedství o poloměru r. Na obrázku 9 můžeme pozorovat jednotlivé shluky, vazby mezi nimi a propojení mezi jednotlivými uzly uvnitř shluků.



Obrázek 10: Neighbourhood Graph 2

Na obrázku 10 můžeme pozorovat výsledné mapy pro různé poloměry. Vzhledem k povaze vstupních dat a pro názornou ukázku NG vizualizace jsou hodnoty parametru r relativně malé. Na obrázku 10(a), kde je parametr $r = 1 \cdot 10^{-17}$, můžeme spatřit hrany, které jsou propojeny jenom mezi některými uzly. Můžeme zaznamenat jenom několik úseček, které nedávají dostatečný náhled na vztahy mezi uzly. Při navýšení parametru $r = 1 \cdot 10^{-16}$ (obrázek 10(b)) je možné pozorovat přibývání úseček. Nejčastěji přibyly úsečky v shlucích a poblíž jejich center. Oproti obrázku 10(a) již pozorujeme náznaky shluků a jejich center a hranic mezi nimi. Při parametru $r = 1 \cdot 10^{-5}$ (obrázek 10(c)) , již snadno pozorujeme jednotlivé shluky a jejich hranice. Na obrázku 10(c) můžeme zaznamenat několik úseček, které nepatří do žádného většího shluku, jedná se o spojení dvou až tří uzlů, kde jejich vzdálenost vůči sobě je malá, ale pro připojení do většího shluku je parametr r nedostatečný. Po navýšení parametru $r = 0,2$ (obrázek 10(d)) můžeme pozorovat, že hranice mezi shluky se stírají a shluky se začaly propojovat. V případě neustálého navyšování parametru r bychom došli až do stádia, kde by všechny uzly byly propojeny mezi sebou.

5 Implementace programu

Kapitola je věnována praktické části programu. Nejprve bude vysvětlen výběr programovacího jazyka. Následně bude znázorněn vzhled grafického rozhraní programu, struktura vstupních dat, programu a napojení jednotlivých částí do jednoho funkčního celku a možnosti jeho rozšíření.

5.1 Výběr programovacího jazyka

Programovací jazyk C# byl zvolen již v zadání práce. Jedná se o vysokoúrovňový objektově orientovaný programovací jazyk, který je součástí platformy Microsoft .NET Framework. Program je postaven na verzi Microsoft .NET Framework 4.0 [10] a od toho se odvíjí minimální požadavky na systém, ve kterém lze spustit program.

Minimální požadavky tedy jsou:

- Procesor: 1 GHz
- RAM: 512 MB
- HDD:
  - x86: 850 MB
  - x64: 2 GB
- Windows XP
- Microsoft .NET Framework 4.0 Client

5.2 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní je rozděleno na tří části. V první části nalezneme hlavní nabídku, která obsahuje základní ovládací prvky pro volbu metody vizualizace či otevření vstupních dat. Druhá část slouží jako vykreslovací plátno, kde se zobrazují výsledné grafické výstupy. V třetí části jsou umístěny ovládací prvky, které ovlivňují kritéria výpočtů vizualizačních metod. Ovládací prvky jsou zpřístupněny podle aktuálně vybrané vizualizační metody. Výsledné GUI lze pozorovat na obrázku 11.

Obrázek 11: Grafické uživatelské rozhraní

5.3 Struktura vstupních dat a programu

5.3.1 Vstupní data

Vstupní data jsou v binární podobě a mají předem danou strukturu uložených informací. Formát vstupního souboru je rozdělen na dvě části. V první části se jedná o binární soubor s příponou „Bbin“. Tento soubor je pouze informativní a jsou v něm uloženy parametry vstupních neuronů. Struktura tohoto souboru vypadá následovně:

- velikost výstupu na ose x
- velikost výstupu na ose y
- počet vah neuronu
- počet opakování
- počet datových souborů
- název souboru

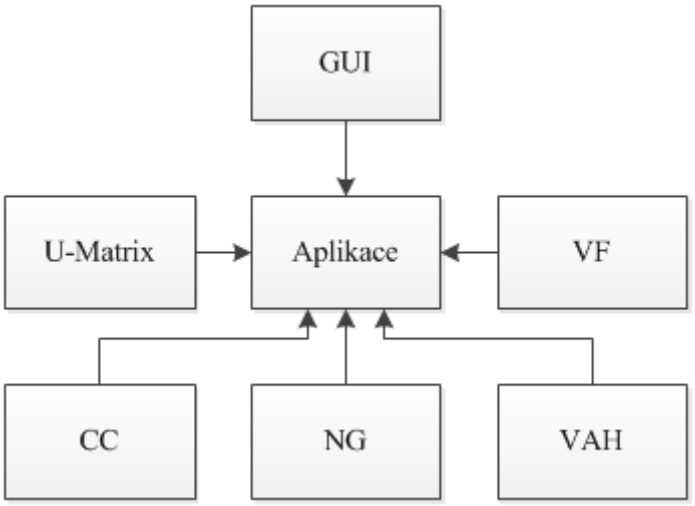
V druhé části se jedná o binární data s příponou „1Bbin“. Jedná se o data, která nesou informace o jednotlivých neuronech naučené SOM. Fyzicky data mohou být rozdělena na více souborů s příponou „1Bbin“. Zda je mapa rozdělená na více binárních souborů, zjistíme z informací, které jsou obsaženy v souboru s příponou „Bbin“. V této

části dat se nachází informace o počtu neuronů. Počet je uložen na prvním řádku v každém souboru těchto dat. Máme-li jediný soubor s příponou „1Bbin“, pak jeho struktura bude vypadat takto:

- počet neuronů
- souřadnice X
- souřadnice Y
- váhy jednoho neuronu

kde souřadnice X, Y a váhy se budou opakovat.

5.3.2 Vazby v programu



Obrázek 12: Vazby v programu

Program je koncipován jako soubor knihoven napojených na jednu centrální část, která se pak kompiluje do spustitelného formátu. Jednotlivé metody jsou kompilovány do dynamických knihoven. Knihovny s metodami nejsou nijak propojeny mezi sebou a to má za následek snadnou přenositelnost jednotlivých metod. Některé metody vyžadují pro svůj správný výpočet vstupní parametry, které dostávají od centrálního prvku, a proto není zapotřebí přímé napojení na GUI, které je řešeno pomocí samostatné knihovny. Metody jsou soběstačné co se týče grafického výstupu, kde se výsledná mapa

vytváří přímo v knihovně a je jenom posílána do centrálního prvku. Pro zachování univerzálnosti knihoven je možno poslat data pro vytvoření grafické mapy v číselném formátu.

Vzhledem k uspořádaní programu je snadné k němu připojit i další metody vizualizací pomoci dalších knihoven.

6 Závěr

Mým cílem v bakalářské práci bylo získat přehled v oblasti grafové vizualizace naučené neuronové sítě typu Self-Organizing Map. Následně získané znalosti pak uplatnit při vytváření aplikace zabývající se grafovou vizualizaci, která využívá různé metody pro zobrazení neuronové sítě. V teoretické části jsem popsal začátky umělých neuronových sítí, kde jsem se zaměřil na typ neuronové sítě SOM. V druhé části práce jsem se snažil podat teoretické informace o jednotlivých metodách a tyto znalosti jsem využil v praktické části bakalářské práce. Jednalo se o pět různých metod (Vector Activity Histogram, U-Matrix, Vector Fields, Cluster Connections a Neighbourhood Graph), kde každá jednotlivá metoda poskytovala jiný pohled na naučenou SOM a vazby mezi neurony.

Možnosti budoucího rozvoje vytvořené aplikace jsou velice rozsáhlé. Aplikaci lze rozšířit o další metody vizualizace, ale zajímavější by bylo aplikaci rozšířit o možnost učení. V případě rozšíření aplikace o možnost učení, kde by vstupními daty byla nenaučená SOM, tak by aplikace mohla sloužit i pro studijní účely, kde by studenti mohli pozorovat celý proces učení nenaučené SOM a následně porovnávat výsledky v různých metodách vizualizace.

7 Reference

[1] ŠÍMA, Jiří a Roman NERUDA, *Teoretické otázky neuronových sítí*. Praha 1996, MATFYZPRESS, ISBN 80-85863-18-9.

[2] MCCULLOCH, Warren a Walter PITTS, A logical calculus of the ideas immanent in nevrous activity. *Bulletin of mathematical biophysics*, 1943

[3] KOHONEN, Teuvo, Self-Organized formation of topologically correct feature maps *Biological Cybernetics 43*, 1982

[4] DEZA, Michel Marie a Elena DEZA *Encyclopedia of distances. 2nd ed.*. New York 2012, Springer, 978-364-2309-571.

[5] ŽÁČEK, Viktor, *Kohonenova samoorganizační mapa*. Brno 2012. Diplomová práce. Vysoké učení technické v Brně, `https://dspace.vutbr.cz/bitstream/handle/11012/12663/xzacek01.pdf`

[6] ULTSCH, Alfred a H. Peter SIEMON. Kohonen's self-organizing feature maps for exploratory data analysis. In: *Proceedings of the International Neural Network Conference*. Kluwer, 1990, s. 305-308.

[7] MERKL, Dieter a Andreas RAUBER. Alternative Ways for Cluster Visualization in Self-Organizing Maps. In: *Proceedings of the Workshop on Self-Organizing Maps*. Finland, 1997, s. 4-6.

[8] POELZLBAUER, Georg, Michael DITTENBACH a Andreas RAUBER. Advanced visualization of Self-Organizing Maps with vector fields. In: *Neural Networks*. 19. vyd. Vienna, Austria, 2006, s. 911-922.

[9] POELZLBAUER, Georg, Michael DITTENBACH a Andreas RAUBER. Advanced visualization techniques for self-organizing maps with graph-based methods. In: *Proceedings of the Second International Symposium on Neural Networks*. Chongqing, China: Springer-Verlag, 2005, s. 75-80.

[10] Microsoft .NET Framework 4. MICROSOFT *Microsoft .NET Framework 4*. [online]. [cit. 2013-04-22].Dostupné z: `http://www.microsoft.com/cs-cz/download/details.aspx?id=17851`

A Obsah CD

- text - text bakalářské práce ve formátu pdf
- aplikace - obsahuje složky source, exe, tests
- aplikace/source - zdrojové kódy
- aplikace/exe - spustitelná aplikace
- aplikace/tests - testovací soubory